Intelligent Differential Quantization of Video Coding

Related Application Information

The following co-pending U.S. patent applications relate to the present application and are hereby incorporated herein by reference: 1) U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Advanced Bi-Directional Predictive Coding of Video Frames," filed concurrently herewith; 2) U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Intraframe and Interframe Interlace Coding and Decoding," filed concurrently herewith; 3) U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Coding of Motion Vector Information," filed concurrently herewith; 4) U.S. Patent Application Serial No. 10/321,415, entitled, "Skip Macroblock Coding," filed December 16, 2002; and 5) U.S. Patent Application Serial No. 10/379,615, entitled "Chrominance Motion Vector Rounding," filed March 4, 2003.

Copyright Authorization

Technical Field

The invention relates generally to differential quantization in digital video coding or compression.

Background

Digital video consumes large amounts of storage and transmission capacity. A typical raw digital video sequence includes 15 or 30 frames per second. Each frame can include tens or hundreds of thousands of pixels (also called pels). Each pixel represents a

tiny element of the picture. In raw form, a computer commonly represents a pixel with 24 bits. Thus, the number of bits per second, or bit rate, of a typical raw digital video sequence can be 5 million bits/second or more.

Most computers and computer networks lack the resources to process raw digital video. For this reason, engineers use compression (also called coding or encoding) to reduce the bit rate of digital video. Compression can be lossless, in which quality of the video does not suffer but decreases in bit rate are limited by the complexity of the video. Or, compression can be lossy, in which quality of the video suffers but decreases in bit rate are more dramatic. Decompression reverses compression.

In general, video compression techniques include intraframe compression and interframe compression. Intraframe compression techniques compress individual frames, typically called I-frames or key frames. Interframe compression techniques compress frames with reference to preceding and/or following frames, which are typically called predicted frames, P-frames, or B-frames.

Microsoft Corporation's Windows Media Video, Version 8 ["WMV8"] includes a video encoder and a video decoder. The WMV8 encoder uses intraframe and interframe compression, and the WMV8 decoder uses intraframe and interframe decompression.

A. Intraframe Compression in WMV8

Figure 1 illustrates block-based intraframe compression 100 of a block 105 of pixels in a key frame in the WMV8 encoder. A block is a set of pixels, for example, an 8x8 arrangement of pixels. The WMV8 encoder splits a key video frame into 8x8 blocks of pixels and applies an 8x8 Discrete Cosine Transform ["DCT"] 110 to individual blocks such as the block 105. A DCT is a type of frequency transform that converts the 8x8 block of pixels (spatial information) into an 8x8 block of DCT coefficients 115, which are frequency information. The DCT operation itself is lossless or nearly lossless. Compared to the original pixel values, however, the DCT coefficients are more efficient for the encoder to compress since most of the significant information is concentrated in low frequency coefficients (conventionally, the upper left of the block 115) and many of

the high frequency coefficients (conventionally, the lower right of the block 115) have values of zero or close to zero.

The encoder then quantizes 120 the DCT coefficients, resulting in an 8x8 block of quantized DCT coefficients 125. For example, the encoder applies a uniform, scalar quantization step size to each coefficient. Quantization is lossy. Since low frequency DCT coefficients tend to have higher values, quantization results in loss of precision but not complete loss of the information for the coefficients. On the other hand, since high frequency DCT coefficients tend to have values of zero or close to zero, quantization of the high frequency coefficients typically results in contiguous regions of zero values. In addition, in some cases high frequency DCT coefficients are quantized more coarsely than low frequency DCT coefficients, resulting in greater loss of precision/information for the high frequency DCT coefficients.

The encoder then prepares the 8x8 block of quantized DCT coefficients 125 for entropy encoding, which is a form of lossless compression. The exact type of entropy encoding can vary depending on whether a coefficient is a DC coefficient (lowest frequency), an AC coefficient (other frequencies) in the top row or left column, or another AC coefficient.

The encoder encodes the DC coefficient 126 as a differential from the DC coefficient 136 of a neighboring 8x8 block, which is a previously encoded neighbor (e.g., top or left) of the block being encoded. (Figure 1 shows a neighbor block 135 that is situated to the left of the block being encoded in the frame.) The encoder entropy encodes 140 the differential.

The entropy encoder can encode the left column or top row of AC coefficients as a differential from a corresponding column or row of the neighboring 8x8 block. Figure 1 shows the left column 127 of AC coefficients encoded as a differential 147 from the left column 137 of the neighboring (to the left) block 135. The differential coding increases the chance that the differential coefficients have zero values. The remaining AC coefficients are from the block 125 of quantized DCT coefficients.

The encoder scans 150 the 8x8 block 145 of predicted, quantized AC DCT coefficients into a one-dimensional array 155 and then entropy encodes the scanned AC

coefficients using a variation of run length coding 160. The encoder selects an entropy code from one or more run/level/last tables 165 and outputs the entropy code.

B. Interframe Compression in WMV8

Interframe compression in the WMV8 encoder uses block-based motion compensated prediction coding followed by transform coding of the residual error. Figures 2 and 3 illustrate the block-based interframe compression for a predicted frame in the WMV8 encoder. In particular, Figure 2 illustrates motion estimation for a predicted frame 210 and Figure 3 illustrates compression of a prediction residual for a motion-estimated block of a predicted frame.

For example, the WMV8 encoder splits a predicted frame into 8x8 blocks of pixels. Groups of four 8x8 blocks form macroblocks. For each macroblock, a motion estimation process is performed. The motion estimation approximates the motion of the macroblock of pixels relative to a reference frame, for example, a previously coded, preceding frame. In Figure 2, the WMV8 encoder computes a motion vector for a macroblock 215 in the predicted frame 210. To compute the motion vector, the encoder searches in a search area 235 of a reference frame 230. Within the search area 235, the encoder compares the macroblock 215 from the predicted frame 210 to various candidate macroblocks in order to find a candidate macroblock that is a good match. After the encoder finds a good matching macroblock, the encoder outputs information specifying the motion vector (entropy coded) for the matching macroblock so the decoder can find the matching macroblock during decoding. When decoding the predicted frame 210 with motion compensation, a decoder uses the motion vector to compute a prediction macroblock for the macroblock 215 using information from the reference frame 230. The prediction for the macroblock 215 is rarely perfect, so the encoder usually encodes 8x8 blocks of pixel differences (also called the error or residual blocks) between the prediction macroblock and the macroblock 215 itself.

Figure 3 illustrates an example of computation and encoding of an error block 335 in the WMV8 encoder. The error block 335 is the difference between the predicted block 315 and the original current block 325. The encoder applies a DCT 340 to the error block

335, resulting in an 8x8 block 345 of coefficients. The encoder then quantizes 350 the DCT coefficients, resulting in an 8x8 block of quantized DCT coefficients 355. The quantization step size is adjustable. Quantization results in loss of precision, but not complete loss of the information for the coefficients.

The encoder then prepares the 8x8 block 355 of quantized DCT coefficients for entropy encoding. The encoder scans 360 the 8x8 block 355 into a one dimensional array 365 with 64 elements, such that coefficients are generally ordered from lowest frequency to highest frequency, which typically creates long runs of zero values.

The encoder entropy encodes the scanned coefficients using a variation of run length coding 370. The encoder selects an entropy code from one or more run/level/last tables 375 and outputs the entropy code.

Figure 4 shows an example of a corresponding decoding process 400 for an inter-coded block. Due to the quantization of the DCT coefficients, the reconstructed block 475 is not identical to the corresponding original block. The compression is lossy.

In summary of Figure 4, a decoder decodes (410, 420) entropy-coded information representing a prediction residual using variable length decoding 410 with one or more run/level/last tables 415 and run length decoding 420. The decoder inverse scans 430 a one-dimensional array 425 storing the entropy-decoded information into a two-dimensional block 435. The decoder inverse quantizes and inverse discrete cosine transforms (together, 440) the data, resulting in a reconstructed error block 445. In a separate motion compensation path, the decoder computes a predicted block 465 using motion vector information 455 for displacement from a reference frame. The decoder combines 470 the predicted block 465 with the reconstructed error block 445 to form the reconstructed block 475.

The amount of change between the original and reconstructed frame is termed the distortion and the number of bits required to code the frame is termed the rate for the frame. The amount of distortion is roughly inversely proportional to the rate. In other words, coding a frame with fewer bits (greater compression) will result in greater distortion, and vice versa.

C. Bi-directional Prediction

Bi-directionally coded images (e.g., B-frames) use two images from the source video as reference (or anchor) images. For example, referring to Figure 5, a B-frame 510 in a video sequence has a temporally previous reference frame 520 and a temporally future reference frame 530.

Some conventional encoders use five prediction modes (forward, backward, direct, interpolated and intra) to predict regions in a current B-frame. In intra mode, an encoder does not predict a macroblock from either reference image, and therefore calculates no motion vectors for the macroblock. In forward and backward modes, an encoder predicts a macroblock using either the previous or future reference frame, and therefore calculates one motion vector for the macroblock. In direct and interpolated modes, an encoder predicts a macroblock in a current frame using both reference frames. In interpolated mode, the encoder explicitly calculates two motion vectors for the macroblock. In direct mode, the encoder derives *implied* motion vectors by scaling the co-located motion vector in the future reference frame, and therefore does not explicitly calculate any motion vectors for the macroblock.

D. Interlace Coding

A typical interlaced video frame consists of two fields scanned at different times. For example, referring to Figure 6, an interlaced video frame 600 includes top field 610 and bottom field 620. Typically, the odd-numbered lines (top field) are scanned at one time (e.g., time t) and the even-numbered lines (bottom field) are scanned at a different (typically later) time (e.g., time $t + 1$). This arrangement can create jagged tooth-like features in regions of a frame where motion is present because the two fields are scanned at different times. On the other hand, in stationary regions, image structures in the frame may be preserved (i.e., the interlace artifacts visible in motion regions may not be visible in stationary regions).

E. Standards for Video Compression and Decompression

Aside from WMV8, several international standards relate to video compression and decompression. These standards include the Motion Picture Experts Group ["MPEG"] 1, 2, and 4 standards and the H.261, H.262, and H.263 standards from the International Telecommunication Union ["ITU"]. Like WMV8, these standards use a combination of intraframe and interframe compression. The MPEG 4 standard describes coding of macroblocks in 4:2:0 format using, for example, frame DCT coding, where each luminance block is composed of lines from two fields alternately, and field DCT coding, where each luminance block is composed of lines from only one of two fields.

F. Differential Quantization

Differential quantization is a technique in which the amount of quantization applied to various blocks within a single video frame can vary. Differential quantization has been adopted or used in various standards. The key benefit is to control bit rate at finer resolution to meet hardware requirements. One common problem that occurs when it is used is that the visual quality is compromised, especially when it is used in low bit rate encoding. For example, signaling quantization parameters individually per each block in a frame of video can consume a significant number of bits in the compressed bitstream, which bits could otherwise be used to encode better quality video.

Given the critical importance of video compression and decompression to digital video, it is not surprising that video compression and decompression are richly developed fields. Whatever the benefits of previous video compression and decompression techniques, however, they do not have the advantages of the following techniques and tools.

Summary

A video compression encoder/decoder (codec) described herein includes techniques for intelligent differential quantization. With these techniques, video can be intelligently quantized at differing strength levels within a frame, such as on a macroblock (MB) or a group of MB basis. The key benefits of intelligent differential

quantization are the abilities to control bit usage on a finer granularity than a frame to meet hardware constraints (e.g., in a CD player, DVD player, etc.). In addition, the intelligent differential quantization allows perceptual optimization by coarsely quantizing unimportant regions, while finely quantizing important regions within a frame.

The intelligent differential quantization techniques are particularly beneficial in consumer devices that have a fixed reading/writing speed requirement, and can not handle a sudden burst of data. By allowing the codec to control the amount of data generated on a finer scale, manufacturers will be able to build consumer devices that can more readily handle the compressed bitstream. In addition, intelligent differential quantization helps to improve the perceptual quality of the video.

The intelligent differential quantization techniques described herein address this quality loss issue. The techniques use the information gathered from encoding and analysis of the video to classify the importance of different regions of the image and quantize them accordingly. In addition, the techniques include an efficient way to signal all the necessary information of the differential quantization strengths in the compressed bit stream.

Additional features and advantages of the invention will be made apparent from the following detailed description of embodiments that proceeds with reference to the accompanying drawings.

Brief Description Of The Drawings

Figure 1 is a diagram showing block-based intraframe compression of an 8x8 block of pixels according to the prior art.

Figure 2 is a diagram showing motion estimation in a video encoder according to the prior art.

Figure 3 is a diagram showing block-based interframe compression for an 8x8 block of prediction residuals in a video encoder according to the prior art.

Figure 4 is a diagram showing block-based interframe decompression for an 8x8 block of prediction residuals in a video encoder according to the prior art.

Figure 5 is a diagram showing a B-frame with past and future reference frames according to the prior art.

Figure 6 is a diagram showing an interlaced video frame according to the prior art.

Figure 7 is a block diagram of a suitable computing environment in which several described embodiments may be implemented.

Figure 8 is a block diagram of a generalized video encoder system used in several described embodiments.

Figure 9 is a block diagram of a generalized video decoder system used in several described embodiments.

Figure 10 is a flow chart of an intelligent differential quantization method in the video encoder/decoder system of Figures 8-9.

Figure 11 is a syntax diagram of a syntax for signaling intelligent differential quantization in the video encoder/decoder system of Figures 8-9.

DETAILED DESCRIPTION

For purposes of illustration, the innovations summarized above are incorporated into embodiments of a video encoder and decoder (codec) illustrated in Figures 8-9, which in one embodiment implements a version of the Windows Media Video codec standard (e.g., the current version 9 of this standard). In alternative embodiments, the innovations described herein can be implemented independently or in combination in the context of other digital signal compression systems, and other video codec standards. In general, the depicted video encoder and decoder incorporating the techniques can be implemented in a computing device, such as illustrated in Figure 7. Additionally, the video encoder and decoder incorporating the techniques can be implemented in dedicated or programmable digital signal processing hardware in other digital signal processing devices.

I. Computing Environment

Figure 7 illustrates a generalized example of a suitable computing environment 7 00 in which several of the described embodiments may be implemented. The computing environment 700 is not intended to suggest any limitation as to scope of use or functionality, as the techniques and tools may be implemented in diverse general-purpose or special-purpose computing environments.

With reference to Figure 7, the computing environment 700 includes at least one processing unit 710 and memory 720. In Figure 7, this most basic configuration 730 is included within a dashed line. The processing unit 710 executes computer-executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory 720 may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory 720 stores software 780 implementing a video encoder or decoder.

A computing environment may have additional features. For example, the computing environment 700 includes storage 740, one or more input devices 750, one or more output devices 760, and one or more communication connections 770. An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment 700. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment 700, and coordinates activities of the components of the computing environment 700.

The storage 740 may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment 700. The storage 740 stores instructions for the software 780 implementing the video encoder or decoder.

The input device(s) 750 may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing environment 700. For audio or video encoding, the input

device(s) 750 may be a sound card, video card, TV tuner card, or similar device that accepts audio or video input in analog or digital form, or a CD-ROM or CD-RW that reads audio or video samples into the computing environment 700. The output device(s) 760 may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment 700.

The communication connection(s) 770 enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video input or output, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

The techniques and tools can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment 700, computer-readable media include memory 720, storage 740, communication media, and combinations of any of the above.

The techniques and tools can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

For the sake of presentation, the detailed description uses terms like "indicate," "choose," "obtain," and "apply" to describe computer operations in a computing environment. These terms are high-level abstractions for operations performed by a

computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

II. Generalized Video Encoder and Decoder

Figure 8 is a block diagram of a generalized video encoder 800 and Figure 9 is a block diagram of a generalized video decoder 900.

The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, Figures 8 and 9 generally do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bit stream, typically after entropy encoding of the side information. The format of the output bit stream can be a Windows Media Video format or another format.

The encoder 800 and decoder 900 are block-based and use a 4:1:1 macroblock format. Each macroblock includes four 8x8 luminance blocks and four 4x8 chrominance blocks. Further details regarding the 4:1:1 format are provided below. The encoder 800 and decoder 900 also can use a 4:2:0 macroblock format with each macroblock including four 8x8 luminance blocks (at times treated as one 16x16 macroblock) and two 8x8 chrominance blocks. Alternatively, the encoder 800 and decoder 900 are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration.

Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques.

A. Video Encoder

Figure 8 is a block diagram of a general video encoder system 800. The encoder system 800 receives a sequence of video frames including a current frame 805, and

produces compressed video information 895 as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder 800.

The encoder system 800 compresses predicted frames and key frames. For the sake of presentation, Figure 8 shows a path for key frames through the encoder system 800 and a path for predicted frames. Many of the components of the encoder system 800 are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed.

A predicted frame (also called P-frame, B-frame, or inter-coded frame) is represented in terms of prediction (or difference) from one or more reference (or anchor) frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame (also called I-frame, intra-coded frame) is compressed without reference to other frames.

If the current frame 805 is a forward-predicted frame, a motion estimator 810 estimates motion of macroblocks or other sets of pixels of the current frame 805 with respect to a reference frame, which is the reconstructed previous frame 825 buffered in a frame store (e.g., frame store 820). If the current frame 805 is a bi-directionally-predicted frame (a B-frame), a motion estimator 810 estimates motion in the current frame 805 with respect to two reconstructed reference frames. Typically, a motion estimator estimates motion in a B-frame with respect to a temporally previous reference frame and a temporally future reference frame. Accordingly, the encoder system 800 can comprise separate stores 820 and 822 for backward and forward reference frames. For more information on bi-directionally predicted frames, see U.S. Patent Application Serial No. aa/bbb,ccc, entitled, "Advanced Bi-Directional Predictive Coding of Video Frames," filed concurrently herewith.

The motion estimator 810 can estimate motion by pixel, ½ pixel, ¼ pixel, or other increments, and can switch the resolution of the motion estimation on a frame-by-frame basis or other basis. The resolution of the motion estimation can be the same or different horizontally and vertically. The motion estimator 810 outputs as side information motion

information 815 such as motion vectors. A motion compensator 830 applies the motion information 815 to the reconstructed frame(s) 825 to form a motion-compensated current frame 835. The prediction is rarely perfect, however, and the difference between the motion-compensated current frame 835 and the original current frame 805 is the prediction residual 845. Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation.

A frequency transformer 860 converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer 860 applies a discrete cosine transform ["DCT"] or variant of DCT to blocks of the pixel data or prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer 860 applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. If the encoder uses spatial extrapolation (not shown in Figure 8) to encode blocks of key frames, the frequency transformer 860 can apply a re-oriented frequency transform such as a skewed DCT to blocks of prediction residuals for the key frame. In some embodiments, the frequency transformer 860 applies an 8x8, 8x4, 4x8, or other size frequency transforms (e.g., DCT) to prediction residuals for predicted frames.

A quantizer 870 then quantizes the blocks of spectral data coefficients. The quantizer applies uniform, scalar quantization to the spectral data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the quantizer applies another type of quantization to the spectral data coefficients, for example, a non-uniform, vector, or non-adaptive quantization, or directly quantizes spatial domain data in an encoder system that does not use frequency transformations. In addition to adaptive quantization, the encoder 800 can use frame dropping, adaptive filtering, or other techniques for rate control.

If a given macroblock in a predicted frame has no information of certain types (e.g., no motion information for the macroblock and no residual information), the encoder 800 may encode the macroblock as a skipped macroblock. If so, the encoder signals the skipped macroblock in the output bit stream of compressed video information 895.

When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer 876 performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer 866 then performs the inverse of the operations of the frequency transformer 860, producing a reconstructed prediction residual (for a predicted frame) or a reconstructed key frame. If the current frame 805 was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame 805 was a predicted frame, the reconstructed prediction residual is added to the motion-compensated current frame 835 to form the reconstructed current frame. A frame store (e.g., frame store 820) buffers the reconstructed current frame for use in predicting another frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

The entropy coder 880 compresses the output of the quantizer 870 as well as certain side information (e.g., motion information 815, spatial extrapolation modes, quantization step size). Typical entropy coding techniques include arithmetic coding, differential coding, Huffman coding, run length coding, LZ coding, dictionary coding, and combinations of the above. The entropy coder 880 typically uses different coding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular coding technique.

The entropy coder 880 puts compressed video information 895 in the buffer 890. A buffer level indicator is fed back to bit rate adaptive modules.

The compressed video information 895 is depleted from the buffer 890 at a constant or relatively constant bit rate and stored for subsequent streaming at that bit rate. Therefore, the level of the buffer 890 is primarily a function of the entropy of the filtered, quantized video information, which affects the efficiency of the entropy coding. Alternatively, the encoder system 800 streams compressed video information immediately following compression, and the level of the buffer 890 also depends on the rate at which information is depleted from the buffer 890 for transmission.

Before or after the buffer 890, the compressed video information 895 can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information 895.

B. Video Decoder

Figure 9 is a block diagram of a general video decoder system 900. The decoder system 900 receives information 995 for a compressed sequence of video frames and produces output including a reconstructed frame 905. Particular embodiments of video decoders typically use a variation or supplemented version of the generalized decoder 900.

The decoder system 900 decompresses predicted frames and key frames. For the sake of presentation, Figure 9 shows a path for key frames through the decoder system 900 and a path for predicted frames. Many of the components of the decoder system 900 are used for decompressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being decompressed.

A buffer 990 receives the information 995 for the compressed video sequence and makes the received information available to the entropy decoder 980. The buffer 990 typically receives the information at a rate that is fairly constant over time, and includes a jitter buffer to smooth short-term variations in bandwidth or transmission. The buffer 990 can include a playback buffer and other buffers as well. Alternatively, the buffer 990 receives information at a varying rate. Before or after the buffer 990, the compressed video information can be channel decoded and processed for error detection and correction.

The entropy decoder 980 entropy decodes entropy-coded quantized data as well as entropy-coded side information (e.g., motion information 915, spatial extrapolation modes, quantization step size), typically applying the inverse of the entropy encoding performed in the encoder. Entropy decoding techniques include arithmetic decoding, differential decoding, Huffman decoding, run length decoding, LZ decoding, dictionary decoding, and combinations of the above. The entropy decoder 980 frequently uses

different decoding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular decoding technique.

A motion compensator 930 applies motion information 915 to one or more reference frames 925 to form a prediction 935 of the frame 905 being reconstructed. For example, the motion compensator 930 uses a macroblock motion vector to find a macroblock in a reference frame 925. A frame buffer (e.g., frame buffer 920) stores previously reconstructed frames for use as reference frames. Typically, B-frames have more than one reference frame (e.g., a temporally previous reference frame and a temporally future reference frame). Accordingly, the decoder system 900 can comprise separate frame buffers 920 and 922 for backward and forward reference frames.

The motion compensator 930 can compensate for motion at pixel, ½ pixel, ¼ pixel, or other increments, and can switch the resolution of the motion compensation on a frame-by-frame basis or other basis. The resolution of the motion compensation can be the same or different horizontally and vertically. Alternatively, a motion compensator applies another type of motion compensation. The prediction by the motion compensator is rarely perfect, so the decoder 900 also reconstructs prediction residuals.

When the decoder needs a reconstructed frame for subsequent motion compensation, a frame buffer (e.g., frame buffer 920) buffers the reconstructed frame for use in predicting another frame. In some embodiments, the decoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

An inverse quantizer 970 inverse quantizes entropy-decoded data. In general, the inverse quantizer applies uniform, scalar inverse quantization to the entropy-decoded data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the inverse quantizer applies another type of inverse quantization to the data, for example, a non-uniform, vector, or non-adaptive quantization, or directly inverse quantizes spatial domain data in a decoder system that does not use inverse frequency transformations.

An inverse frequency transformer 960 converts the quantized, frequency domain data into spatial domain video information. For block-based video frames, the inverse

frequency transformer 960 applies an inverse DCT ["IDCT"] or variant of IDCT to blocks of the DCT coefficients, producing pixel data or prediction residual data for key frames or predicted frames, respectively. Alternatively, the frequency transformer 960 applies another conventional inverse frequency transform such as a Fourier transform or uses wavelet or subband synthesis. If the decoder uses spatial extrapolation (not shown in Figure 9) to decode blocks of key frames, the inverse frequency transformer 960 can apply a re-oriented inverse frequency transform such as a skewed IDCT to blocks of prediction residuals for the key frame. In some embodiments, the inverse frequency transformer 960 applies an 8x8, 8x4, 4x8, or other size inverse frequency transforms (e.g., IDCT) to prediction residuals for predicted frames.

When a skipped macroblock is signaled in the bit stream of information 995 for a compressed sequence of video frames, the decoder 900 reconstructs the skipped macroblock without using the information (e.g., motion information and/or residual information) normally included in the bit stream for non-skipped macroblocks.

III. Intelligent Differential Quantization

With reference to Figure 10, the video encoder 800/decoder 900 described above implements intelligent differential quantization techniques in a process 1000 that intelligently quantizes/dequantizes at differing strength levels within a frame, such as on a macroblock (MB) or a group of MB basis. The techniques use the information gathered from encoding and analysis of the video to classify the importance of different regions of the image and quantize/dequantize them accordingly.

More particularly, the video encoder 800/decoder 900 analyzes the global motion of the video to classify the importance of the regions within a frame. As discussed above, the video encoder 800 gathers motion vector information in the encoding process, which is used in encoding the video (e.g., for predictive interframe coding). This motion vector information is encoded as side information in the compressed bit stream. Based on the motion vector information gathered in the encoding process, the video encoder 800/decoder 900 estimates the global motion of the video (at action 1010), including whether the video is panning left/right/up/down/diagonals or zooming in/out.

In one embodiment, the video panning detection can be performed be calculating an aggregate value of the motion vectors within the video frame, and comparing this aggregate value to a motion threshold value. If the aggregate motion vector exceeds the threshold, the video is determined to be panning in the opposite direction. Zoom detection in some embodiments of the invention can be performed by calculating an aggregate of the motion vectors for separate quadrants of the video frame, and testing whether the quadrants' aggregate motion vectors are directed inwardly or outwardly. In alternative embodiments, other methods of video panning and zoom detection based on the motion vectors can be used.

Based on this global motion estimate, the intelligent differential quantization technique then classifies which regions of the video frame may be less important to perceptual quality of the video (action 1020). In particular, if the video is panning toward some direction, the opposite side of the image has less perceptual significance, and can be more coarsely quantized without much impact of overall perceptual quality. For example, if the video is panning towards left, then the right edge of the image will quickly disappear in the following frames. Therefore, the quality of the disappearing edge macroblocks can be compromised (compressed more) to save bits to either meet the bit rate requirement or to improve quality of other part of images without much perceptual degradation. Likewise, if the video is zooming in, the all edges of the image will quickly disappear in the following frames, and the quality of all these disappearing edge macroblocks can be compromised.

According to the intelligent differential quantization technique, the video encoder 800 determines the differential quantization to apply to macroblocks in the frame at action 1030. The regions classified as less perceptually significant are quantized more strongly, which saves bits that can be used to meet bit rate requirements or to decrease the quantization of the macroblocks in regions that are not classified as less perceptually significant.

At action 1040, the video encoder 800 encodes information in the compressed bit stream using a signaling scheme described below for signaling the differential quantization to the video decoder 900. At decoding, the video decoder 900 reads the

signaled differential quantization information, and dequantizes the macroblocks accordingly to decompress the video.

A. Differential Quantization Signaling Scheme

With reference to Figure 11, the video encoder 800 encodes information for signaling the differential quantization that was applied in compressing the video to the video decoder 900. In one embodiment, the video encoder 800 encodes side information in the compressed bit stream using a syntax of the Windows Media Video (WMV) standard. This syntax structure is described in part in the co-filed patent application entitled, "Coding Of Motion Vector Information," which is incorporated herein by reference above; and in the patent application entitled, " Skip Macroblock Coding," filed 12/16/2002, which also is incorporated herein by reference above. In alternative embodiments, the syntax in which the information used in the intelligent differential quantization can be modified for use in another video compression standard or video coding scheme.

Figure 11 depicts the syntax structure of the side information sent in the compressed bit stream for intelligent differential quantization in this embodiment of the video encoder 800. This side information includes information at the sequence, frame and macroblock levels of the syntax. As further detailed below, the syntax can represent for individual frames in a video sequence, the different quantization applied to macroblocks in respective regions of the frame or to each macroblock individually. The syntax can represent that different quantization levels are applied to macroblocks on each of the frame's boundary edges, to pairs of adjacent boundary edges, to all boundary edges, or all macroblocks individually. This permits the syntax to efficiently signal the various regions identified for differential quantization as being less perceptually significant due to panning and/or zooming.

On sequence header (which is sent per video sequence), this syntax includes a DQUANT flag 1120, which is a 2-bit field that indicates whether or not the quantization step size can vary within a frame. In this syntax, there are three possible values for DQUANT. If DQUANT = 0, then only one quantization step size (i.e. the frame

quantization step size) is used per frame. If DQUANT = 1 or 2, the DQUANT flag indicates the possibility to quantize each macroblock in the frame differently.

On the frame level, a VOPDQUANT field 1110 is made up of several bitstream syntax elements as shown in Figure 11. The VOPDQUANT field is present in Progressive P picture and Interlace I and P pictures in the sequence, when the sequence header DQUANT field is nonzero. The syntax of the VOPDQUANT field is dependent on the picture type (whether it's an I picture or a P picture) and the value of the DQUANT flag, as follows.

Case 1: DQUANT = 1.

In this case, the syntax provides four possibilities:

1. The macroblocks located on the boundary are quantized with a second quantization step size (ALTPQUANT) while the rest of the macroblocks are quantized with the frame quantization step size (PQUANT).

2. The encoder signals two adjacent edges (per Table 1 below) and those macroblocks located on the two edges are quantized with ALTPQUANT while the rest of the macroblocks are quantized with PQUANT.

3. The encoder signals one edge and those macroblock located on the edge are quantized with ALTPQUANT while the rest of the macroblocks are quantized with PQUANT.

4. Every single macroblock can be quantized differently. In this case, we will indicate whether each macroblock can select from two quantization steps (PQUANT or ALTPQUANT) or each macroblock can be arbitrarily quantized using any step size.

Case 2: DQUANT = 2.

The macroblocks located on the boundary are quantized with ALTPQUANT while the rest of the macroblocks are quantized with PQUANT.

The bitstream syntax for case 1 includes the following fields:

- **DQUANTFRM** (1 bit)
  The DQUANTFRM field 1131 is a 1 bit value that is present only when DQUANT = 1. If DQUANTFRM = 0 then the current picture is only quantized with PQUANT.

- **DQPROFILE** (2 bits)
  The DQPROFILE field 1132 is a 2 bits value that is present only when DQUANT = 1 and DQUANTFRM = 1. It indicates where we are allowed to change quantization step sizes within the current picture. This field is coded to represent the location of the differentially quantized region as shown in the code Table 1 below.

Table 1: Macroblock Quantization Profile (DQPROFILE) Code Table

FLC	Location
00	All four Edges
01	Double Edges
10	Single Edges
11	All Macroblocks

- **DQSBEDGE** (2 bits)
  The DQSBEDGE field 1133 is a 2 bits value that is present when DQPROFILE = Single Edge. It indicates which edge will be quantized with ALTPQUANT, as shown in the following Table 2.

Table 2: Single Boundary Edge Selection (DQSBEDGE) Code Table

FLC	Boundary Edge
00	Left
01	Top
10	Right
11	Bottom

- **DQDBEDGE** (2 bits)

The DQSBEDGE field 1134 is a 2 bits value that is present when DQPROFILE = Double Edge. It indicates which two edges will be quantized with ALTPQUANT, as shown in the following code Table 3.

Table 3: Double Boundary Edges Selection (DQDBEDGE) Code Table

FLC	Boundary Edges
00	Left and Top
01	Top and Right
10	Right and Bottom
11	Bottom and Left

- **DQBILEVEL** (1 bit)

  The DQBILEVEL field 1135 is a 1 bit value that is present when DQPROFILE = All Macroblock. If DQBILEVEL = 1, then each macroblock in the picture can take one of two possible values (PQUANT or ALTPQUANT). If DQBILEVEL = 0, then each macroblock in the picture can take on any quantization step size.

- **PQDIFF** (3 bits)

  The PQDIFF field 1136 is a 3 bit field that encodes either the PQUANT differential or encodes an escape code.

  If the PQDIFF field does not equal 7 then the PQDIFF field encodes the differential and the ABSPQ field does not follow in the bitstream. In this case:

  $$ALTPQUANT = PQUANT + PQDIFF + 1$$

  If the PQDIFF field equals 7 then the ABSPQ field follows in the bitstream and the ALTPQUANT value is decoded as:

  $$ALTPQUANT = ABSPQ$$

- **ABSPQ** (5 bits)

  The ABSPQ field 1137 is present in the bitstream if PQDIFF equals 7. In this case, ABSPQ directly encodes the value of ALTPQUANT as described above.

In view of the many possible embodiments to which the principles of our invention may be applied, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.